Image Modelling with Position-Encoding Dynamic Trees

Amos J Storkey and Christopher K I Williams

Institute for Adaptive and Neural Computation

Division of Informatics, University of Edinburgh

5 Forrest Hill, Edinburgh UK

{*a.storkey,c.k.i.williams*}*@ed.ac.uk*

June 20, 2002

Abstract

This paper describes the Position-Encoding Dynamic Tree (PEDT). The PEDT is a probabilistic model for images which improves on the Dynamic Tree by allowing the positions of objects to play a part in the model. This increases the flexibility of the model over the Dynamic Tree and allows the positions of objects to be located and manipulated.

The paper motivates and defines this form of probabilistic model using the belief network formalism. A structured variational approach for inference and learning in the PEDT is developed, and the resulting variational updates are obtained, along with additional implementation considerations which ensure the computational cost scales linearly in the number of nodes of the belief network. The PEDT model is demonstrated and compared with the dynamic tree and fixed tree. The structured variational learning method is compared with mean field approaches.

Keywords: dynamic trees, variational inference, belief networks, bayesian networks, image segmentation, structured image models, tree structured networks.

1 Introduction

Consider an observer who is presented with a road-scene image. She is likely to be interested primarily in "what is in the image", which can be taken to mean identifying components such as trees, cars, road, sky and clouds. This can be thought of as finding a label for every pixel such that there is a coherent interpretation of the scene.

Taking a Bayesian view of this problem, we can separate it into two parts, a scene model and a pixel model. The scene model defines a distribution over label fields; this model should incorporate information about properties such as the spatial coherence of objects and their likely locations in the image. The pixel model relates the scene model to the observed image data. The labelling problem thus becomes one of *inference*, i.e. to infer the posterior distribution over label fields given an observed image, or some summary of this distribution, such as the *maximum a posteriori* (MAP) configuration.

There are many possible models for label fields. Some examples include Markov random fields (MRFs) and and tree-structured belief networks with a fixed structure. See section 2 for further discussion of these models. In this paper we discuss an alternative hierarchical model which we term the Position Encoding Dynamic Tree (PEDT). This is based on the idea that labelling an image is a two dimensional analogue of creating a parse-tree for a sentence, so that the tree-structure must reflect the underlying image structure; this does not happen with fixed-structure tree-structured belief networks (TSBNs). The hierarchical structure of the PEDT also provides the capacity to represent part—sub-part relationships.

This paper makes a number of contributions. First, we present a dynamic tree hierarchical image model which configures itself in response to a given image, where the nodes in the hierarchical model have both positional and state information. Second, we show how the model parameters can be learnt for a particular set of segment labels and set of images. Third, we discuss inference in such a network and derive a *structured variational* approximation. Finally we give demonstrations on a number of image sets, and provide comparisons with related methods.

The structure of the remainder of the paper is as follows: Section 2 discusses work related to the topic of this paper. Section 3 gives a motivation for the position encoding dynamic tree, which is introduced and defined in section 4. A discussion of methods for inference in the PEDT model is the topic of section 5, leading to the inferential updates of section 6. Learning the parameters of the model is discussed in section 7. The model is demonstrated and tested on various datasets in section 8, before conclusions are drawn. The appendices give the calculation of the update equations, and methods for comparing dynamic trees and position encoding dynamic trees.

2 Related work

We first describe related work which follows a Bayesian formulation of the problem, and then discuss other related work.

Within the Bayesian framework there are a number of popular models for label images, the most popular of which are the MRF and TSBN models. In the statistical image modelling community these two types of model are known as non-causal and causal MRF models respectively. They are respectively undirected and directed graphical models [17].

Early work on Bayesian image modelling concentrated on non-causal MRFs, see. e.g. [4, 12]. Note that these models typically have a "flat", non-hierarchical structure. They also suffer from high computational complexity, for example the problem of finding the *maximum a posteriori* (MAP) interpretation given an image is (in general) NP-hard.

The alternative causal MRF formulation uses a directed graph, and the most commonly used form of these models is a tree-structured belief network. In contrast to MRFs, TSBNs provide a hierarchical multiscale model for an image. They also have efficient inference algorithms which run in time linear in the number of nodes in the tree. In the graphical models literature this inference procedure is known as Pearl's message passing scheme [22]. This algorithm is also known as the upward-downward or inside-outside algorithm [24, 18], being a generalisation to trees of the standard Baum-Welch forward-backward algorithm for HMMs. TSBNs with discrete-valued nodes have

been used for image segmentation tasks [5, 23]. TSBN models have also been used for continuously-valued Gaussian processes in one and two dimensions, see for example the work of Willsky's group at MIT [3, 20, 19].

Despite these attractive properties, TSBNs are not the ultimate image model. Run generatively, fixed-structure TSBNs give rise to "blocky" images which reflect the structure of the underlying tree. One idea to move beyond TSBNs is to remove the tree-structured constraint, so that a child node depends on more than one parent. Exploration of this idea includes the work of Bouman and Shapiro [5] on a cross-linked architecture, and Dayan et al. [8] on the Helmholtz machine. One problem is that exact inference has a considerably higher time complexity in non-tree structures (one needs to use the junction tree algorithm, see e.g. [17]).

An alternative view is that the problem with TSBNs is not the tree structure, but the fact that it is a *fixed* tree. It is reasonable that for opaque objects each object-part belongs to at most one object, so that there is a "single-parent constraint" (see [15] for discussion of this point). This suggests that a model should provide a distribution over tree structures, reminiscent of parse-trees obtained with context-free grammars (see e.g. [6]). Our work builds on that of Adams and Williams on "dynamic trees" [29, 1] and Hinton et al. [15] on "credibility networks".

TSBN and Dynamic Tree models are hierarchical multiscale models, as are wavelet models. For example Crouse [7] have used a multiscale TSBN to model wavelet coefficients, and DeBonet and Viola [9] have used an interesting tree-structured network for image synthesis using non-Gaussian densities. However, note that as we require a prior over class-label images, we cannot use the *additive* combination of contributions from different levels used in wavelet modelling.

A general probabilistic view of the labelling problem can be found in [16]; the key point is that the labelling predictions are not independent at every pixel, so it is a *contextual* classification problem.

We would like to point out some relationships to other work which is not expressed

in the probabilistic modelling framework. Von der Malsburg [27, 28] has discussed the Dynamic Link Architecture, whereby the network architecture changes dynamically in response to the input. This parallels the inference process in DT architectures, where posterior tree structures are effectively selected in accordance with how well they fit the image structure. We also note that Montanvert et al [21] have discussed irregular tree-structured networks, where the tree-structure is image dependent.

We would also like to mention the similarities and differences of the labelling problem with the image segmentation problem. Image segmentation techniques aim to divide up images into homogeneous regions; they are often based on region-growing or edge-based techniques, or combinations of both. Note that the key difference between the labelling and segmentation problems is that segmentation is essentially an unsupervised learning problem (e.g. a spatially coherent clustering of image features) while image labelling is based on a supervised learning problem (learning image labels from features).

The formalism described here can also be described in terms of multinets [11] which are a generalisation of Bayesian networks to include context sensitive dependency structures. In multinets, different belief network structures for a set of random variables X are allowed depending of the value of some context variable Z.

3 Motivation

Any form of image analysis must use, implicitly or explicitly, prior information about how images are formed. However sometimes this prior information is used in an ad-hoc way. The Bayesian approach aims to make the prior information explicit through the use of representative models and probability distributions over those models. To follow a Bayesian approach for image segmentation a good probabilistic framework for the prior knowledge must be found. Then the Bayesian machinery will provide consistent inference.

What sort of prior information can be used about the images? Usually an image contains a number of regions corresponding to different objects (used in a broad sense of the term). The existence of these objects provides significant prior information. Furthermore

each object is made of different parts, which can each be seen as objects at another scale (for example a face is made of eyes, nose mouth etc.). A natural way of modelling objects would involve a hierarchical representation, where the objects lower in the hierarchy are related in some way to those immediately above. This is the approach which is followed in the development of position encoding dynamic trees.

The model developed here takes its impetus from a generative approach. Instead of primarily looking for features or characteristics within an image, and then trying to use this information for segmentation, recognition or some other purpose, a generative approach starts with asking what is known about how the image came about. This gives some prior knowledge of what might be expected to be seen in an image. This prior knowledge is used to build what is called a *generative model*. The perceived image is then used to refine that prior knowledge to provide a reasonable model (or set of models) for *that* particular image.

Given that images can be seen as being constructed from many different objects, each with subcomponents and further substructures, it would appear sensible to model objects hierarchically. But a simple deterministic model will not capture the variability in object structure between different images or parts of images. The same object type might have significant variation between different images, in terms of position, colour, lighting etc, but also in terms of the subcomponents it might have. It seems then that a probabilistic model is more appropriate to capture this variation and the magnitude of the variation.

Using a tree structured directed graph is one way to define a hierarchical probabilistic model. In such a structure, nodes are used to represent the different variables, and the probability of a node being in some state is given in terms of the possible states of its parent (each directed edge of the graph goes from a *parent* node to a *child* node). In other words whether the parent is in a certain state or not affects the state probability of the child node. This seems reasonable and is related to the way that parse tree structure is often used for the one dimensional problem of sentence modelling. Any dependence between objects would only occur through the component-subcomponent relationship.

Hence distinct unrelated objects would be probabilistically independent in the model.

The problem with using a fixed tree structure is that the inherent organisation of different object scenes would not be correctly represented by the same hierarchical relationships. It is important to be able to represent the variability in the subcomponents an object might have. The dynamic tree framework [29, 2, 26] is a significant step in that direction. Even so, there are problems associated with the dynamic tree structure. Although there is no longer a restriction to a single tree structure, the model is still non-stationarity. The same image shifted slightly could have significantly different maximum posterior representations. Furthermore finding the location of an object is non-trivial, and using the model in sequences is hard, as movement has to correspond to a change in connectivity. The position-encoding dynamic tree model which is introduced here has a richer representation which, it is argued, overcomes these deficits.

4 Position-encoding Dynamic Trees

An image can be thought of as containing a number of objects or regions each of which we might choose to label in a way which relates to the nature of the object. For example we might want to label all people in an image with the same label, all cars with another label, all trees with another and so on. Or we might choose to label certain types of surface or texture one way and other surfaces/textures other ways etc. These regions will tend to be spatially contiguous in some way, and in defining models for object labels of images this spatial structure should be taken into account. Labels will not contain all the object information we need. For example we would wish to distinguish one person from another person. We will see that the connectivity of the posterior PEDT is able to do this.

The PEDT model uses a hierarchical tree structure of nodes. Each node has a label denoting the type of object (or object part) which it is representing. It also has a position which represents the spatial location of the object part being represented. These labels could just be the labels we expect to get at the pixel level, but might also include

additional latent labels used to represent higher level structures. Each node represents an object or part of an object, and the position of the node gives an idea of the region of the image the node is representing. Nodes further up the tree represent larger regions of the image. This representation is not direct but is given by the influence that each node has on the labels of its children, down to the lowest level of nodes which are taken to correspond to individual pixel labels (or labels for specific groups of pixels). This influence is defined in terms of a conditional probability of the label of the child node given that we know the label of the parent node.

Each image will have a different tree structure associated with it, and a different set of labels and positions for each node (in fact because of the uncertainty in what would be the best tree structure, the PEDT approach will obtain a probability distribution over the structure, positions and labels for each image). In order to generate these different structures and labels for each image we need to specify and to learn what are reasonable possible tree structures for images and reasonable labellings. To do this a prior probability distribution is defined and the parameter values of that prior are learnt from training images.

This PEDT model will give us a number of benefits. First of all it utilises a model of image labels which will improve on any pixelwise labelling of an image by using the spatial structure to tailor the label probabilities. Secondly it gives us a structural model of the relationship between different parts of the image, and lastly it gives the positions associated with the different parts or objects in an image which would then be useful for use in sequence models. Lastly because of the choice of the form of the PEDT model, highly efficient, significantly structured, variational methods can be used to perform inference and learn parameters [26, 13]. This makes (approximate) inference in the PEDT model reasonably efficient.

4.1 The Position-encoding dynamic tree model

We are now in a position to give the overall generative model for the position-encoding dynamic tree. There are four fundamental parts to it, the network nodes, the dynamic structure, the node classes and the node positions. Here we will introduce the basic structure, but other variables can be introduced if wanted without major modification.

The model will be described using the belief network formalism. The graphical model will have a number of layers of nodes which will be used to describe objects or object parts, each layer representing object parts of similar sizes. The 'lower' nodes will represent smaller components and the 'higher' nodes larger ones.

The set N of n network nodes is denoted by $\{1, 2, \ldots, n\}$. The nodes are organised into layers $\{1, 2, \ldots, H\}$, where 1 denotes the top layer. We use L^h to denote the set of nodes in layer h. The network nodes are connected together in a way that represents the structural relationships in the image. We generally use the superscript notation to represent a set of random variables. For example X^B represents the set $\{X_i | i \in B\}$. For the state of all nodes we drop the N: $X^N = X$. We will also use the shorthand X' to denote X^{L^B} where $B = \{1, 2, \ldots, H-1\}$. In other words X' represents the set of X_i corresponding to all nodes i not in the bottom layer.

The network connectivity is denoted by a matrix Z of indicator variables z_{ij}. Setting $z_{ij} = 1$ represents the existence of the connection between child i and parent j. We also include the variable z_{i0}, which is set to 1 if and only if i is a root node (i.e. it has no parent). Because Z must be a tree structure, i can have at most one parent, and so given i, $z_{ij} = 1$ for one and only one value of j and is zero otherwise. Furthermore, the constraint that all nodes can only have parents in the layer above means z_{ij} can only equal 1 if node j is in a layer one higher than node i (or $j = 0$).

The variable Z can represent all the possible tree structures which are of interest. It is now possible to define a prior distribution over tree connectivity, which is taken to be

of the form

$$P(Z) = \prod_{i=1,j=1}^{n} \gamma_{ij}^{z_{ij}}, \tag{1}$$

where γ_{ij} is the probability of i being the child of j (we will say i chooses parent j with probability γ_{ij}). Generally it is assumed that this probability is uniform across all parents $j \neq 0$, expressing node interchangeability. Interchangeability is not a theoretical necessity (and so a more general form of γ_{ij} could be used). However, because we will allow each node to take any position, interchangeability does make the later computational optimisations easier as it introduces a symmetry which means that any one of the symmetric optima can be found, rather than one particular optimum. The disconnection probability γ_{i0} needs to be specified apriori.

4.2 Positions

Each node has a value which represents the position of the centre of the object. The position of each node is given relative to the position of the parent node, or if the node has no parent the position is taken to be absolute. The prior distribution over each of these nodes needs to be specified.

For the root nodes, we suppose the position is taken from a broad Gaussian distribution. Gaussian distributions are used to specify the centres of the child objects given the parent. Formally[1]

$$P(\mathbf{r}_i|\mathbf{r}_j, z_{ij} = 1) = \frac{1}{(2\pi)^{d/2}|\Sigma_{ij}|^{1/2}} \exp(-\frac{1}{2}(\mathbf{r}_i - \mathbf{r}_j)^T(\Sigma_{ij})^{-1}(\mathbf{r}_i - \mathbf{r}_j)), \tag{2}$$

where $d = 2$ is the dimensionality of the space, and Σ_{ij} is a covariance used to represent the order of magnitude of the object size.

[1] It is also possible to include offests ρ_{ij} within this formalism, so that $P(\mathbf{r}_i|\mathbf{r}_j, z_{ij} = 1)$ takes the form $(2\pi)^{-d/2}|\Sigma_{ij}|^{-1/2} \exp(-\frac{1}{2}(\mathbf{r}_i - \mathbf{r}_j - \boldsymbol{\rho}_{ij})^T(\Sigma_{ij})^{-1}(\mathbf{r}_i - \mathbf{r}_j - \boldsymbol{\rho}_{ij}))$. They are left out here for the sake of simplicity.

The overall distribution is given by

$$P(R|Z) = \prod_{ij|z_{ij}=1} \frac{1}{(2\pi)^{d/2}|\Sigma_{ij}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{r}_i - \mathbf{r}_j)^T(\Sigma_{ij})^{-1}(\mathbf{r}_i - \mathbf{r}_j)\right). \tag{3}$$

At the lowest level (layer H) we will need to connect the nodes up to the pixels. For this reason we set the positions of the bottom layer of node to be given by the pixel positions, and then use the distribution $P(Z, R'|R^{L^H})$ as the prior over positions and connectivity.

4.3 The node labels

The other remaining important concept is the class label of each of the nodes. This represents the type of object or texture which the node is representing.

Suppose there are C possible classes which nodes can belong to. We denote the class of a node i by X_i and use the indicator $x_i^k = 1$ to represent the state that X_i is in class k. We represent the probability of a node i being in state k given its parent j is in state l by a conditional probability table P_{ij}^{kl}. We make a simplifying assumption that this conditional probability is independent of the positions of the parent and child nodes. This is not entirely realistic, but is a useful first approximation. Hence $P(X|Z)$ can be written as

$$P(X|Z) = \prod_{ijkl} (P_{ij}^{kl})^{x_i^k x_j^l z_{ij}}. \tag{4}$$

4.4 Pixel intensities

The intensity of each pixel will depend on the class label of the node it is connected to. For each bottom layer node i there is a pixel in position $\mathbf{r}_i$. Let this pixel also be enumerated by i, and assume its value is entirely dependent on the node state X_i through the distribution $P(Y_i|X_i)$, where Y_i denotes the pixel RGB intensity. The overall pixel

model is

$$P(Y|X,R) = \prod_{i \in L^H} P(Y_i|X_i). \tag{5}$$

The simple form of pixel model used in the experiments in this paper represents $P(Y_i|X_i)$ by the histogram of class conditional pixel intensities over all the pixels in the training data.

4.5 All together

The full definition of the prior distribution is given by the following. First $P(Z)$ is defined using (1) and $P(R|Z)$ using (3) to give us $P(R,Z)$. We then impose the condition fixing the bottom layer nodes to the pixel positions to get $P(Z,R'|R^{L^H})$. Using equation (4) gives $P(X,R',Z|R^{L^H}) = P(Z,R'|R^{L^h})P(X|Z)$. Lastly $P(Y|X)$ is of the form (5), giving the final joint distribution $P(Y,X,R',Z|R^{L^H})$. This fully specifies a position encoding dynamic tree model.

5 Variational inference

The position-encoding dynamic tree model gives the prior probability distribution for the image through the use of a large hierarchical latent variable model. To use it, we need to find out what the posterior probability distribution of the latent variables is given a specific image. In other words we condition on the fact that we have pixels Y, and try to discover the distribution over Z, X and R' that results. This distribution will give us a set of possible good interpretations of the connectivity, content and positioning of objects in the image.

Finding the posterior of the PEDT model is non-trivial, and so some approach for doing Bayesian inference is needed. Were this model a tree structured belief network, we could use techniques such as belief propagation [22] to exactly calculate the required posterior distribution $P(X,R',Z|Y,R^{L^h})$. However because the PEDT is not of this form, and because other approaches such as the junction tree method are inappropriate because of the large clique size of the triangulated network, exact calculations are not

feasible. Instead the graphical structure of the distribution is used to develop variational approximation methods to the posterior.

Variational methods are one of a number of approaches for approximate inference in networks. Other approaches include Markov chain Monte-Carlo methods [14], where a Markov chain is constructed which has the desired posterior as the limit distribution. Samples from this Markov chain are used as an approximation for the posterior. Monte-Carlo and annealing approaches for dynamic trees were investigated in [29]. Unfortunately obtaining Monte-Carlo samples for a network of this size would take longer than is affordable. The variational approach, on the other hand, fits an approximating distribution to the true posterior. The variational distribution must be chosen to be tractable to calculate with and must allow a variational fit to the posterior to be obtained in a reasonable time. At the same time the fit must be as good as possible so that the approximation is not a poor one. This makes choosing the variational distribution a non-trivial exercise. Further details of the variational method can be found in [25].

After an outline of the variational method and the form of variational distribution which is used, the resulting update equations used to fit the distribution to the posterior is given in section 6. The full calculations can be found in the appendix A.

The variational approach of this section develops and extends the approach used in [26] to the new case of the position-encoding dynamic tree. This approach involves approximating the posterior distribution with a factorising distribution of the form $Q(Z)Q(R)$ $Q(X|Z)$, where $Q(Z)$ is the approximating distribution over the Z variables, $Q(X|Z)$ is the approximating distribution over the states, and $Q(R)$ is an approximating distribution over the node positions.

To choose good forms for the Q's the Kullback-Leibler divergence between the $Q(Z)$ $Q(R)Q(X|Z)$ distribution and the true posterior should be minimised. In fact the approximate distributions which are used take the form of a dynamic tree model, and give propagation rules which are efficient and local.

The KL divergence between the approximation and the true posterior is of the form

$$\int dR' \sum_{Z,X} Q(Z)Q(X|Z)Q(R') \log \left(\frac{Q(Z)Q(X|Z)Q(R')}{P(Z,R',X|R^{L^H},Y)} \right). \tag{6}$$

The forms of each of the approximating distributions needs to be finalised. We use a $Q(Z)$ and $Q(R')$ of the form

$$Q(Z) = \prod_{ij} \alpha_{ij}^{z_{ij}} \text{ and } Q(R') = \prod_{i \in N \setminus L^H} \frac{1}{(2\pi)^{d/2}|\Omega_i|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{r}_i^T - \boldsymbol{\mu}_i^T)\Omega_i^{-1}(\mathbf{r}_i - \boldsymbol{\mu}_i) \right) \tag{7}$$

where the α's are probabilities and where $\boldsymbol{\mu}_i$ and Ω_i are position and covariance parameters respectively, all of which need to be optimised. In this paper Ω_i is assumed to be diagonal. Lastly the $Q(X|Z)$ is a dynamic tree distribution of a form identical to that used in [26]:

$$Q(X|Z) = \prod_{ijkl} (Q_{ij}^{kl})^{x_i^k x_j^l z_{ij}}. \tag{8}$$

Again Q_{ij}^{kl} are parameters to be optimised.

This distribution is chosen because local update propagations can be obtained through KL divergence minimisation. Hence the model can be tuned to the posterior efficiently. Also the marginal values of this distribution can be obtained straightforwardly. In addition this variational form is guaranteed to give better approximations than simpler ones such as a mean field approximation [26]. The difference between this approach and the mean field approach stems from the fact that we allow dependence on Z in (8), and that we also allow dependence of a node i on parent nodes rather than forcing $Q(X|Z)$ to take a factorised form. Because there are significant dependencies between child and parent in the prior we would expect some dependence to remain in the posterior. The mean field approach ignores that dependence, whereas this more structured approximating distribution allows some of those dependencies to be captured.

6 Update Equations

We want to minimise the KL divergence (6), with the forms of approximate distribution given in the last section. This must be done subject to the constraints that $\sum_k Q_{ij}^{kl} = 1$ and $\sum_j \alpha_{ij} = 1$ (probabilities sum to 1). We add to (6) a set of Lagrange multiplier terms corresponding to these constraints, and set the derivatives to zero. Solving this gives a set of update equations. The full derivations are given in appendix A. Below, the update equations for the conditional probability tables, position and tree connectivity of the variational distribution are given.

6.1 Class Labels

Minimising the KL divergence gives us a set of update equations. Given all the α's, let m_i^k be given recursively from the top down by

$$m_i^k = \sum_{jl} \alpha_{ij} Q_{ij}^{kl} m_j^l. \tag{9}$$

Then m_i^k is the marginal probability of node i being in class k under the variational distribution. Again given the α's we find that minimisation of the KL divergence gives

$$Q_{ij}^{kl} = \frac{P_{ij}^{kl} \lambda_i^k}{\sum_{k'} P_{ij}^{k'l} \lambda_i^{k'}} \text{ where } \lambda_i^k = \prod_{c \in c(i)} \left[\sum_g P_{ci}^{gk} \lambda_c^g \right]^{\alpha_{ci}}, \tag{10}$$

where $c(i)$ is used to denote the possible children of i, in other words the nodes in the layer below that containing node i.

Hence given α all the Q's can be updated by making a single pass up the tree to calculate the λ values, and then calculating the Q's.

6.2 Positions

The update equations for the positions (again given the α's) take the following forms

$$\boldsymbol{\mu}_i = \sum_k \alpha_{ki}(\Sigma_{ki})^{-1}\boldsymbol{\mu}_k + \sum_j \alpha_{ij}(\Sigma_{ij})^{-1}\boldsymbol{\mu}_j, \tag{11}$$

$$(\Omega_i)_{pp} = \frac{1}{\sum_j \left(\alpha_{ij}(\Sigma_{ij})^{-1}_{pp} + \alpha_{ji}(\Sigma_{ji})^{-1}_{pp}\right)} \tag{12}$$

where we have assumed that both Σ and Ω are diagonal. The equations for $\boldsymbol{\mu}$ involve a sum across all possible parents and a sum across all possible children, and hence need to be iterated until suitably converged.

6.3 Connectivity

Lastly the connectivity needs to be considered. For fixed parameters in $Q(X|Z)$ and $Q(R')$ of the forms given above, we obtain

$$\begin{aligned} \alpha_{ij} &\propto \gamma_{ij}\exp(-\Psi_{ij})\exp(-\Phi_{ij}) \\ \text{where} \quad \Psi_{ij} &= \sum_l m_j^l[\log\sum_k P_{ij}^{kl}\lambda_i^k] \\ \text{and} \quad \Phi_{ij} &= \frac{1}{2}(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)^T\Sigma_{ij}^{-1}(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j) + \frac{1}{2}\mathrm{Tr}(\Sigma_{ij}^{-1}\Omega_i) + \frac{1}{2}\mathrm{Tr}(\Sigma_{ij}^{-1}\Omega_j). \end{aligned} \tag{13}$$

In the above the constant of proportionality is found by normalisation as $\sum_j \alpha_{ij} = 1$.

6.3.1 Highest Variational Posterior Probability

The tree Z which maximises $Q(Z)$ after the variational distributions have been optimised is called the tree with the highest variational posterior probability (HVPP). It is effectively the maximum a posteriori (MAP) tree under the variational approximation. Likewise we can pick the HVVP solution for any set of random variables, such as the classification labels or positions. This is useful for visualisation.

6.4 Optimisation Process

The above equations give all the necessary update rules. The whole optimisation process involves an outer loop optimising the $Q(Z)$ values and an inner loop containing up and down passes of the $Q(X|Z)$ optimisation and a number of passes of the $Q(R)$ optimisation. The KL divergence can be calculated up to an additive constant, and so can be used as an explicit objective function and be monitored accordingly. A termination criterion based on the change in the KL divergence can be set. However the calculation of the KL divergence itself is one of the most costly parts of the optimisation, so in the experiments we used a fixed number of iterations, and only calculated the KL divergence at the end of each inferential run. The whole optimisation takes the following form:

```
Initialise Q(R)
Optimise Q(Z) ignoring Q(X|Z) contribution
for outerloop=1 to numouterloop
  Optimise Q(X|Z)
  for innerloop=1 to numinnerloop
    for qroptim=1 to numqroptim
      Do one pass of the Q(R) optimisation
    end
    Optimise Q(Z)
  end
end
```

In the experiments reported in section 8 we used 7 outer loop iterations, each containing an optimisation of $Q(X|Z)$ and 4 inner loop iterations of the optimisation of $Q(Z)$ and $Q(R)$. The $Q(R)$ optimisation itself involved 3 passes through all the nodes. $Q(R)$ was initialised using the quadtree-like structure described in appendix B. The first optimisation of $Q(Z)$ initialises it to the best variational fit of $Q(R)Q(Z)$ to $P(Z, R'|R^{L^H})$.

6.5 Efficiency issues

There are a few hidden problems in the optimisation. Most of the updates are inexpensive, however there is the issue of summing over all possible children/parents. For each node, a sum over all the nodes in the layer above is required. As this needs to be done for every node, this is very expensive. Most of the elements of this sum will give negligible contributions because their contribution contains a factor from the tail of a Gaussian probability distribution.

To reduce this computational burden, for each layer l, we segment the image space into a grid of boxes, where the length and width of each box is given by the standard deviation of the prior Gaussian distribution $P(\mathbf{r}_i|\mathbf{r}_{Pa(i)})$ of the nodes in layer l. Then for each node i, we identify the box in the parental layer which the position of node i falls within, and only consider prospective parents within the 3×3 subgroup of boxes surrounding and including that box. Other nodes will have a negligible contribution to any of the calculations needed. This is significantly more efficient than considering all possible nodes, and only involves a small number of prospective parents for each node. We thereby reduce the number of references to z_{ij} components which are irrelevant. This keeps these computations down to a small constant times the number of nodes (about $9n$ in a quadtree initialised PEDT for example).

Computing the KL divergence at every step is a computationally expensive procedure, and so although it would be ideal to test the change in the KL divergence for suitable convergence, it is generally more efficient to pre-estimate the number of loops generally needed for suitable convergence, and either not test the KL divergence or test it at the end.

7 Learning

As it stands, the position-encoding dynamic tree model presumes the knowledge of certain parameters. For example it presumes the form of the conditional probability tables P_{ij}^{kl}, and the form of the covariance matrices Σ_{ij} and the model $P(Y_i|X_j)$. Because

the covariance matrices are effectively used to represent a priori object size, they can be specified in a reasonable way to represent the sizes required at each level. In our experiments these variances were set by hand to ensure that after conditioning on R^{L^h} there was a non-negligible probability of each node connecting to a small (about 7-9) number of parent nodes. The conditional probabilities, on the other hand, depend on the inter-relationships of objects in scenes, and would be hard to specify accurately. Instead we can choose to learn the conditional probabilities (and the root node probabilities) using labelled image data. Likewise we can specify a form of model $P(Y_i|X_i)$, and learn that from the labelled images. Because the variational method gives a lower bound to the log likelihood, parameters can be learnt by maximising this lower bound.

7.1 Learning the conditional probability tables P_{ij}

If a labelled segmented dataset of images is available, this can be used to learn the conditional probability tables (CPTs). The simplest approach assumes that the conditional probability tables are in fact all copies of one table. The theory for more flexible possibilities is straightforward. For example one could choose to use a layerwise approach; that is to have different conditional probability tables (CPTs) for each scale (i.e. each layer), as was used for the TSBN model in [30], and is used in section 8. We will outline the theory for the single CPT model, and explain how the results differ for the layerwise approach.

The process for learning the CPTs is as follows. First the conditional probabilities are initialised. Then the variational inference is run for each image given the labels of the leaf node positions. Introducing Θ to represent the conditional probability table parameters, we define

$$\begin{aligned} L_{var}(Y|\Theta) &\stackrel{\text{def}}{=} -\int dR' \sum_{Z,X} Q(Z)Q(R')Q(X|Z) \log \frac{Q(Z)Q(R')Q(X|Z)}{P(Z,R,X,Y|\Theta)} \\ &= -KL_\theta(Q||P) + \log P(Y|R^{L^H},\Theta) + \log P(R^{L^H}) \end{aligned} \tag{14}$$

and use the fact $P(Z,R,X,Y|\Theta) = P(Z|\Theta)P(R|Z,\Theta)P(X|Z,\Theta)P(Y|X,\Theta)$. $L_{var}(Y|\Theta)$ is called the variational log likelihood. The fact that the KL divergence $KL(Q||P) \geq 0$ implies the log likelihood $\log P(Y|R^{L^H},\Theta)$ is lower bounded by the variational log likelihood minus the additive constant $\log P(R^{L^H})$. Repeatedly maximising the variational log likelihood by adapting the Q distributions and then the parameters Θ, will maximise a lower bound to the log probability.

In fact we have more than one image. Let the images be denoted by $Y^1, Y^2, \ldots, Y^m$. Then we maximise a lower bound to the log likelihood by maximising $\sum_{s=1}^{m} L_{var}(Y^s|\Theta)$ which can be done by adapting the Q for each Y^s and then adapting the parameters Θ to maximise the sum of the variational log likelihoods for all the Y^s:

$$\sum_{s=1}^{m} L_{var}(Y^s|\Theta) = -\sum_{s=1}^{m} \int dR' \sum_{Z,X} Q_s(Z)Q_s(R')Q_s(X|Z) \log \frac{Q_s(Z)Q_s(R')Q_s(X|Z)}{P(Z,R,X,Y^s|\Theta)}. \tag{15}$$

Substituting in to (15) for all the terms and taking derivatives of the variational log likelihood with respect to P^{kl} (the kl entry of the shared CPTs) gives

$$P^{kl} \propto \sum_{s=1}^{m} \sum_{ij} (\alpha_s)_{ij} (Q_s)^{kl}_{ij}, (m_s)^j_j \tag{16}$$

where the constant of proportionality is given by normalisation. In the case that the CPTs are stored layerwise, the sum over ij in (16) is replaced by a sum over i in the relevant layer, and over prospective parents j of nodes i.

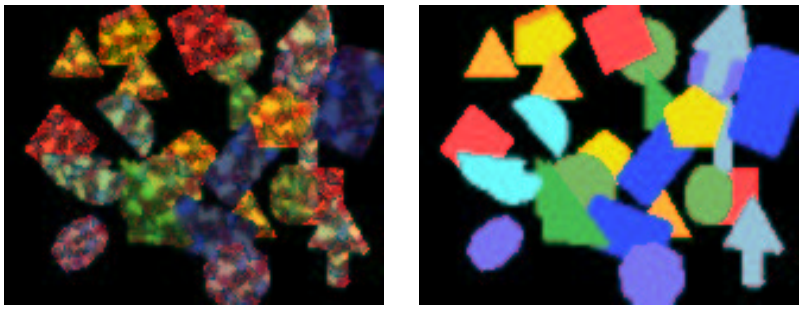
7.2 Learning the pixel model

The pixel model can also be learnt from labelled images. The image provides the Y_i information, and the labels the corresponding X_i values. This can be used to train a neural network [30], a class conditional model [5, 10] or some other suitable model. The Y_i can be pixels, groups of pixels or some region based preprocessing of the pixels. One of the simplest forms of pixel model uses an empirically derived class-conditional model, obtained through sectioning the RGB colour cube of pixel space into smaller regions

and building class-conditional histograms of each of those regions; this approach is used below.

8 Experiments

The PEDT model was tested on a number of artificial ray traced images using the class-conditional histogram pixel model given in section 7.2. Ray traced images have the benefit of being reasonably realistic, but at the same time making ground truth segmentations easy to obtain. Different sets of images were used. The first (henceforth GEOM) was a set of 50 training and 50 test colour images of size 160 by 120 pixels. Each image consisted of a number of patterned 3D objects which are thickened 2D geometric shapes. The shapes are coloured and textured (an example can be seen in figure 1). Ground truth segmentations using 11 labels were provided. 9 of these labels referred to each of the different shapes, one to the background and the remaining label to the side panel of one of the shapes. A $4 \times 4 \times 4$ image cube was used in the pixel model.



(a) (b)

Figure 1: Two images from the GEOM dataset. (a) gives an image from the training set, while (b) gives the corresponding ground truth labels. Different object positions and lighting were used between and within the test and training datasets.

The second artificial image set (henceforth SUNSET) consisted of sunset scenes with labels for each of *sky, water, sun, cloud, helicopter*. There were 6 training images and 6 test images, again all were at 160×120 pixels. The training data consists of a number of different lighting scenarios, angles and positions. Some of the images do not contain any objects of certain classes. For example there might be no helicopter or no sea in the

picture. A $4 \times 4 \times 4$ image cube was used in the pixel model.

In subsection 8.1, the use of the model is demonstrated, illustrating the pixelwise HVPP segmentations (that is the bottom layer labels where each label is chosen with the highest variational posterior probability) obtained from the model at different scales, and also giving some illustration of the HVPP tree structures which are obtained. The model is compared quantitatively with both the structured variational dynamic tree and exact fixed tree approaches in subsection 8.2. In subsection 8.3 the approximation methods are compared with less structured mean field approaches.

The most obvious benefits of the PEDT model are qualitative: the mobility of the nodes and the dynamic architecture allow the shapes to be fitted better than earlier approaches, the blocky segmentation effects are gone, and curves are well approximated. The positions also make temporal modelling possible. The tree structures give some indication of the structural elements of the objects in the picture. As an indicator of the run time, a PEDT inference run on one image of size 160×120, using 11 classes takes about a minute on a 1Ghz PC.

8.1 Demonstrations

C++/MATLAB Position encoding dynamic tree software is available at http://www.anc.ed.ac.uk/code/storkey/. The software takes training and test image sets in most formats and outputs matlab files of the results. There is also an interactive graphical display which allows the user to scroll through the images, network layers and to produce tree slice projections for the vertical and horizontal lines through any chosen point.

Here we demonstrate the performance of the position-encoding dynamic tree on the GEOM and SUNSET datasets. For each dataset, the PEDT was trained on the relevant training set so as to learn the CPTs $P(X_i|X_{Pa(i)})$. The CPTs were assumed to be the same for all nodes in a given layer. The standard deviations of the Gaussian distributions for each layer of the Gaussian model $P(R|Z)$ were set by hand to be of a suitable width: one which generally gave a few (9 or so) possible choices of parent for a node with varying

but non-negligible probabilities. The variational inference algorithm was run on the test set and the results reported below.

When using the model in general the number of training examples needs to be chosen to allow each class to be fairly represented in the different conditions which it might be found in later test images.

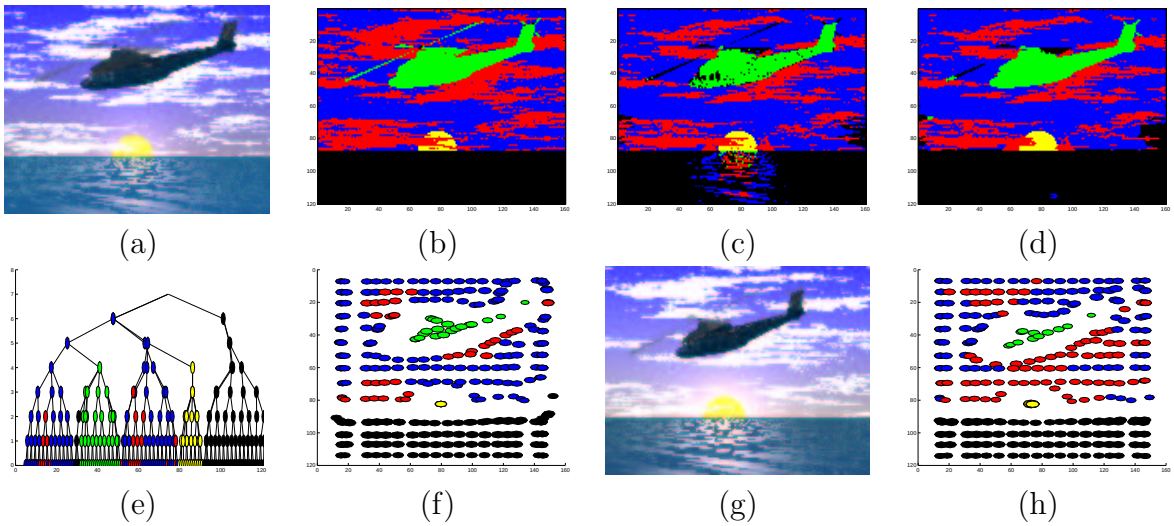


Figure 2: A test example. (a) The image, and (b) the ground truth labelling. (c) The pixelwise labelling and (d) the positional dynamic tree labelling. (e) A slice projection of the highest posterior dynamic tree (from down the middle of the image) and (f) the positions and labels of the sixth layer of the tree. (g) gives the next image in the sequence, while (h) gives shows where the nodes in the sixth layer move to in a subsequent image.

8.1.1 Description

Looking first at the SUNSET data, in figure 2a we have one test image along with the ground truth segmentation (2b). The pixelwise segmentation without the use of the dynamic positional tree can be seen in figure 2c. The picture in 2d gives the pixelwise HVPP segmentation obtained using the variational approach on the dynamic positional tree. This picture only gives a crude idea of the overall posterior distribution. Figure 2e gives a projection of a slice of the HVPP tree structure obtained (we actually have a distribution over trees), while figure 2f gives a picture of the positions and labels of nodes in the sixth layer from the root (out of nine) of the posterior dynamic positional tree.

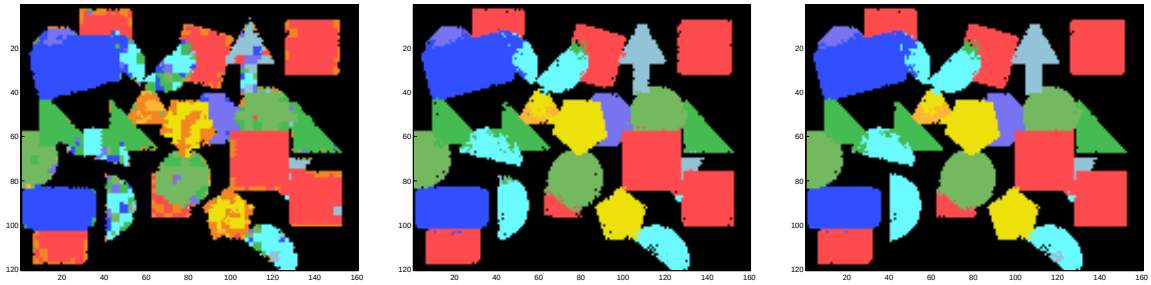
Given a second image (figure 2g) in sequence with the first, we can see what happens to the node positions for a subsequent image in figure 2h.

8.1.2 Comparison with dynamic and fixed tree on the GEOM dataset

The segmentations produced by the PEDT were compared with the fixed tree and dynamic tree. Figure 3 shows the results. The dynamic tree was run using the approach described in appendix B using the same initialisation as the PEDT. There are clear improvements for both the DT and the PEDT over the fixed tree. The improvement of the PEDT over the DT is marginal. However looking at the classification probabilities shows the PEDT is more certain of its classification than the DT in regions where it is correct, and less certain in regions where it is wrong. This is evident if we calculate pixelwise classification probabilities: the average over all test images and all pixels of the log probability of the true pixel classification is -0.0344 for the fixed tree, -0.0281 for the DT and is the highest for the PEDT at -0.0132. Using the mean field algorithm (for inference only) instead of the structured variational method produces comparable results (-0.0138), which is to be expected as the main benefit of the structured variational calculation is the accuracy of the distribution over tree structures. In order to compare the mean field and structured variational inference methods this calculation is carried out using the conditional probabilities learned using the structured variational method.

The results above are equivalent to geometric average classification probabilities of 0.9662, 0.9723, 0.9869 and 0.9863 respectively. These figures are also related to the average cost $E(-\log_2 P(\mathbf{x}))$ of coding the pixel labels given the colour image[2]. This coding cost is about 0.050 bits/pixel for the fixed tree, 0.041 bits/pixel for the dynamic tree and 0.019 bits/pixel for the PEDT (0.020 with mean field inference).

[2]Here we have calculated the predicted pixelwise marginal probabilities of the labels given the colour images and used them to code the label images. The true coding cost will be less than this as our calculation ignores correlations between the predicted label probabilities.

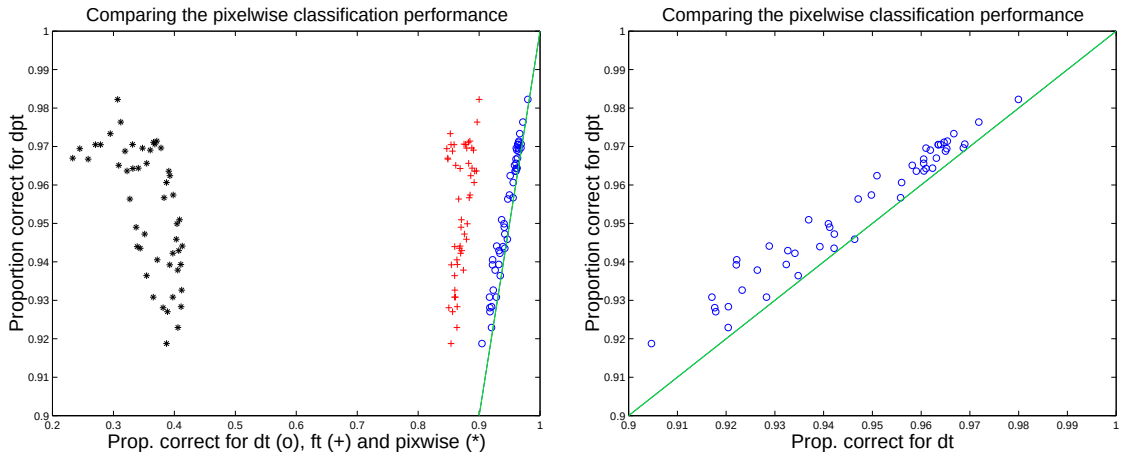


(a) (b) (c)

Figure 3: The segmentation using the (a) the fixed tree (b) the dynamic tree and (c) the PEDT.

8.2 Comparison with other probabilistic tree-based methods: labelling performance

The fixed quadtree approach [29][2], the dynamic tree approach [30, 10] and the position encoding dynamic tree approach were compared in terms of their performance on the GEOM image set, along with a simple pixelwise classification (labelled pixwise in figure 4). Again the dynamic tree was run using the approach described in appendix B using the same initialisation as the PEDT. Also disconnection probabilities were set to zero in all models to ensure a fair comparison. Each model was trained independently on the fifty training images and tested on the fifty test images. Pixelwise labelling performance comparisons were made between all these methods, and the simple approach of using pixelwise prediction. The results are given in figure 4. We see that the position encoding dynamic tree gives significant benefit over a fixed tree approach, and a slight benefit over the dynamic tree. Because the major benefit of the position encoding dynamic tree lies in the richness of the hidden layer representation, we would not expect a major improvement in the labelling results between dynamic trees and the PEDT. Even so we see that some benefit is gained. The benefits of all the methods over the straight pixelwise classification approach is clear. In terms of average performance there are also some improvements. The average percentage correct for the pixelwise approach was 35.7%. For the fixed tree this rises to 87.1%. The dynamic tree and PEDT improve this further to 94.6% and 95.4% respectively.



(a) (b)

Figure 4: (a) Comparison of the pixelwise classification performance of all the methods on the GEOM dataset. The PEDT performs better than the other approaches, although the improvement over the straight dynamic tree is marginal. (b) gives an enlargement of a portion of (a) showing the comparison between the DT and the PEDT. Points on the straight lines indicate equal performance.

8.3 Comparison of approximation method

One of the developments described in this paper is the structured variational approach for various forms of dynamic tree. Here the structured variational approach is compared with with the mean field approach. Because the speed of the mean field calculations are much slower than the structured variational method, it is infeasible to do a full comparison on the datasets given. Furthermore the benefits are best illustrated in a one dimensional situation.

Because the dynamic tree formalism is directly related to the discrete variables of the PEDT approach (see appendix B for how the PEDT can be viewed as a generalised dynamic tree) we can focus on the dynamic tree and assess the benefits of the approximation within that framework.

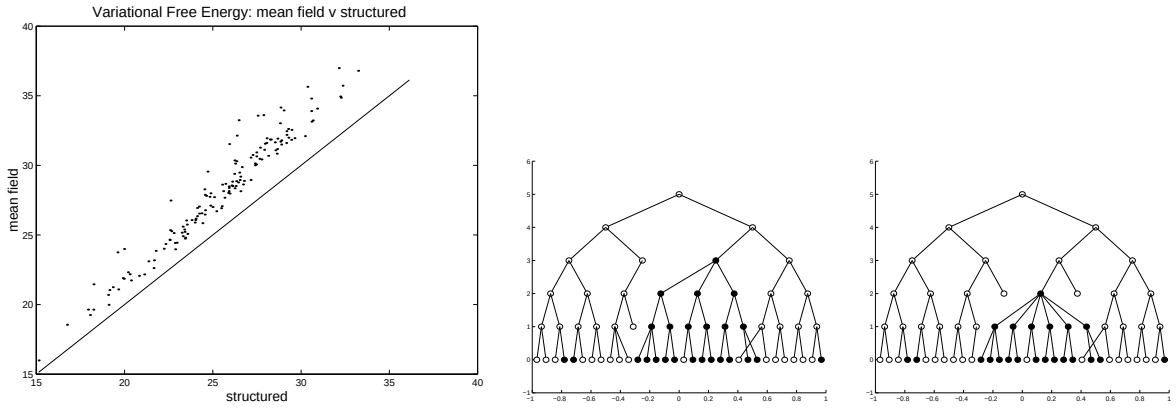
In order to compare the structured variational approach with the mean field method, tests were done using a simple 6 layer, one dimensional dynamic tree, where we represent two possible node states by the colours black and white. 150 cases of one dimensional data were independently generated from a simple Markovian process, and independent

noise was added. This enables us to do a test of how well the model deals with simple locally correlated structures, even if the data might not be easily generated from the prior. For data generated from the prior very similar results are obtained. Conditional probability tables with 0.9 on the diagonal and 0.1 off the diagonal were used throughout.

The dynamic tree implementation structured the prior $P(Z)$ so that each node has a high probability of connecting to the nearest three parents, with a smaller probability of connecting further afield. The implementation of the mean field method was similar to that in [2] with 20 iterations of the $Q(X)$ optimisation for each recalculation of $Q(Z)$. Here $Q(X)$ is the mean field approximation over the class labels X. For the structured variational approach, 5 passes were made through the update procedure. Because the Q values can be calculated exactly in one pass given the α values, convergence of the structured variational approach works out to be significantly faster than the mean field (convergence was assumed if the variational log likelihood changed by less than 0.01 on the current step).

To compare the mean field and structured variational methods, we compare the variational free energy (that is the negative variational log likelihood) found by each method and the true distribution. A lower variational free energy implies a lower KL divergence, and hence a better match to the true posterior. The results of this are shown in figure 5a and indicate that the structured variational method gives a better approximation to the posterior than the mean field approach. Note that the mean field result is a special case of the structured variational approximation. If the structured distribution has $Q_{ij}^{kl} = m_i^k$ for all j, l, it is identical the mean field distribution. When the structured variational approach is optimised, however, we find that the Q do not resemble this degenerate form. Instead they tend to be highly diagonal, and are therefore incorporating some of the conditional dependencies between the nodes.

It is interesting to see what types of structures are produced by the different methods. Figure 5 gives an example of the highest variational posterior probability (HVPP) tree under the approximating distributions for both the mean field and the structured varia-



(a) (b) (c)

Figure 5: (a) Comparison of the variational free energy for the structured variational and mean field approaches. Comparison of the highest probability trees found by the (b) structured variational approximation, and (c) the mean field approximation.

tional approaches for an example input. Note the choice of HVPP tree is for illustration purposes only. We have (and want) posterior *distributions* over trees.

The HVPP trees for the two methods are comparable, but not always identical. For the mean field method there are problems with spontaneous symmetry breaking, which cause the higher level nodes to polarise to one or other state. This has the appearance of 'flattening out' the tree structures relating to the other state variables in the highest posterior tree (see [2] for more details). This effect can be seen in figure 5c. The nodes in the third layer from the top are all dominated by a single class. This prevents the tree structures relating to the 'black' nodes from utilising this layer. This effect is not apparent for the structured variational approximation (figure 5b), and is a possible reason for fact that higher posterior trees are found by it. We would therefore conclude that there are both quantitative, qualitative and computational benefits in the use of the structured variational method.

9 Discussion

The position-encoding dynamic tree is a further step along the road to generic models for images. The model includes a flexible hierarchical representation of image structure, and allows the position information of the structure to be obtained jointly with the class and

structural information. The structured variational method proposed is a better and faster posterior approximation than the mean field approach, giving conditional information for the class labels. The fact that the method is also linear in the number of nodes means that this and related approaches are practical for use with images.

The pixel labelling problem discussed above is germane to a number of real world applications, for example land use classification from remote sensing images. As described here the PEDT approach provides sophisticated methods for combining spatial information with pixelwise class predictions, but the emphasis is still on obtaining class predictions based on some combination of localised pixel features. However there are limitations of any method which uses some local model to map pixels or regions to class labels. For larger numbers of classes, predicting an object type from colour and textural characteristics alone is difficult. In the PEDT framework hierarchical information can help for compound objects, but is only useful if the lowest level labels produced by the pixel model are reasonably accurate in the first place. In problematic situations it could be more suitable to move to more unsupervised methods, using any ground truth labels at a later stage of processing.

The PEDT framework could be adapted for unsupervised methods by allowing the X variables to represent certain types of textural characteristics and building object structures from there, or perhaps better still replacing class labels X with some real valued texture and colour features X^* and using some form of real-valued Gaussian distribution $P(X^*|Z)$ instead. This may also provide advantages in temporal models by giving better clues for correspondence between nodes in two time frames.

The position-encoding dynamic tree formalism introduced here is flexible in that it can include other discrete or real valued variables of interest. It can be used to represent simple dynamic tree and fixed tree methods as well as the PEDT. It jointly deals with the variables of interest to make sure the combined benefits of information from all domains is used together. The PEDT should be seen as a step towards a structured hierarchical object-based image model, where various object characteristics such as type, position,

shape and component structure go to build a generative model for images.

Acknowledgements

This work is supported through EPSRC grant GR/L78161 *Probabilistic Models for Sequences*. We also thank British Aerospace for their support.

A Full derivation of structured variational method

We are interested in choosing $Q(Z)$, $Q(X|Z)$ and $Q(R)$ of the forms (7) through to (8) to minimise the KL divergence (6) (equivalent to maximising the variational log likelihood L_{var} defined in equation (14)). This is tackled in two stages. First we presume we are given $Q(Z)$, and look to optimise $Q(R)$ and $Q(X|Z)$. It turns out that the optimisation of each of these can be done independently. Then given $Q(X|Z)$ and $Q(R)$, we optimise $Q(Z)$.

A.1 Optimisation of $Q(X|Z)$

The components of L_{var} which depend on $Q(X|Z)$ are given by

$$L_X = -\sum_{Z,X} Q(Z)Q(X|Z)\log Q(X|Z) + \sum_{Z,X} Q(Z)Q(X|Z)\log P(X|Z) + \sum_{X,Z} Q(Z)Q(X|Z)\log P(Y|X); \quad (17)$$

all the other components sum out to a constant w.r.t $Q(X|Z)$. Substituting in for $Q(Z)$, $Q(X|Z)$, $P(Y|X)$ and $P(X|Z)$, and taking expectations over z_{ij} gives

$$L_X = -\sum_{ijkl} \alpha_{ij} Q_{ij}^{kl} m_j^l (\log Q_{ij}^{kl} - \log P_{ij}^{kl}) + \sum_{i \in L^H, l} m_i^l \log P(Y_i | x_i^l = 1) \quad (18)$$

where m_i^k is the marginal probability that node i is in state k. This probability is fully determined by all the Q_{ij}^{kl} terms. To calculate the Q_{ij}^{kl} which minimises this L_X we use

$$\frac{\partial L_X}{\partial Q_{ij}^{kl}} = -\frac{\partial V_{ij}}{\partial Q_{ij}^{kl}} - \sum_b \frac{\partial W_i^b}{\partial m_i^b} \frac{\partial m_i^b}{\partial Q_{ij}^{kl}} \quad (19)$$

where

$$V_{ij}(Q_{ij}, \alpha_{ij}, \mathbf{m}_j) = \sum_{kl} \alpha_{ij} \left[Q_{ij}^{kl} m_j^l \log \frac{Q_{ij}^{kl}}{P_{ij}^{kl}} \right] \tag{20}$$

and

$$W_i^k(m_i^k, \{\alpha_{st}, Q_{st} | s \in d(i)\}) = \sum_{s \in d(i), t} \sum_{ab} \alpha_{st} [Q_{st}^{ab} m_t^b (\log Q_{st}^{ab} - \log P_{st}^{ab})] - \sum_{s \in L^H, l \in C} m_s^l \log P(Y_s | x_s^l = 1). \tag{21}$$

Here $d(i)$ denotes the set of nodes in layers below i (i.e. the possible descendants of i). The m_t^b terms are entirely dependent on the m_i^k and the Q's.

Most of the derivatives in (19) are straightforward to compute. The exception is $\frac{\partial W_i^b}{\partial m_i^b}$. This can be obtained by propagating the derivatives from the layer below.

$$\frac{\partial W_i^b}{\partial m_i^b} = \sum_{c \in c(i)} \left(\frac{\partial V_{ci}}{\partial m_i^k} + \sum_p \frac{\partial W_c^p}{\partial m_c^p} \frac{\partial m_c^p}{\partial m_i^k} \right) \tag{22}$$

where $c(i)$ denotes the set of nodes in the layer immediately below i (i.e. possible children of i). The Q's are updated from the bottom layer to the top layer by first propagating the derivatives (22):

$$T_i^k = \frac{\partial W_i^k}{\partial m_i^k} = \sum_{c \in c(i)} \sum_g \alpha_{ci} Q_{ci}^{gk} (\log \frac{Q_{ci}^{gk}}{P_{ci}^{gk}} + T_c^g) \tag{23}$$

defined for $i \in N \setminus L^H$, and $T_i^k = \log P(Y_i | x_i^k = 1)$ for i in L^H. Then optimising $\frac{\partial L_X}{\partial Q_{st}^{ab}}$ with Lagrange multipliers to encode the probabilistic constraints on the Q's gives

$$Q_{ij}^{kl} = \frac{P_{ij}^{kl} \exp(-T_i^k)}{\sum_a P_{ij}^{kl} \exp(-T_i^k)} \tag{24}$$

for all nodes i. Defining λ_i^k to be $\exp(-T_i^k)$, and substituting the above form for Q in to (23), we obtain equation (10) for all $i \in N \setminus L^H$. For $i \in L^H$ we have the initial values $\lambda_i^k = \exp(-T_i^k) = P(Y_i | x_i^k = 1)$. Propagating these λ values up the network allows us

to find all the conditional probabilities Q_{ij}^{kl}. These in turn can be used to calculate the marginal probabilities at each node giving $m_s^k = \sum_t \alpha_{st} \sum_l Q_{st}^{kl} m_t^l$, which is equation (9) If we write $m_s^k = \kappa_s^k \pi_s^k \lambda_s^k$, then we obtain from (24) the equation $\pi_s^k = \sum_{tl} \mu_{st} P_{st}^{kl} \pi_t^l \lambda_t^l / \lambda_{st}^l$ where λ_{ij}^b is given by $\kappa_i \sum_k P_{ij}^{kb} \lambda_i^k / \kappa_j$.

This result is very similar to belief propagation in trees. The whole Q distribution and marginals m (given the α values) can be calculated in two passes. The λ values are propagated up the network and this gives the Q distribution. Then the means m can be propagated down the network. In the special case of $\alpha_{ij} = 1$ for only one value of j, and $\alpha_{ij} = 0$ otherwise (this defines a simple tree structured belief network) the above algorithm reduces to Pearl belief propagation. This can be seen by noticing that in Pearl propagation the constants of proportionality are given by $\kappa_i = \kappa_j = P(Y)$, the probability of the evidential nodes, and by observing that with the above assumption about α,

$$\lambda_{ij}^b = \sum_k P_{ij}^{kb} \lambda_i^k \text{ and } \lambda_i^b = \prod_j \lambda_{ij}^b, \quad \text{and hence} \quad \pi_s^a = \sum_{tb} P_{ij}^{ab} \pi_j^b \prod_{k \neq j} \lambda_{ik}^b. \tag{25}$$

These are the standard belief propagation rules for a tree-structured belief network.

A.2 Optimisation of $Q(R')$

Once again, we need to maximize the variational log likelihood, this time with respect to $Q(R')$. The terms in the variational log likelihood relevant to this optimisation are denoted by L_R and given by

$$L_R = -\langle \log Q(R') \rangle_{Q(R')} - \sum_{ij} \alpha_{ij} \Big[\langle (\mathbf{r}_i - \mathbf{r}_j) \Sigma_{ij}^{-1} (\mathbf{r}_i - \mathbf{r}_j) \rangle_{Q(R')} \Big]. \tag{26}$$

In order to move further we need to make more assumptions about the form of the $Q(R')$ distribution. We choose to use a mean field approximation:

$$Q(R') = \prod_i \frac{1}{(2\pi)^{d/2} |\Omega_i|^{1/2}} \exp\left(-\frac{1}{2} (\mathbf{r}_i^T - \boldsymbol{\mu}_i^T) \Omega_i^{-1} (\mathbf{r}_i - \boldsymbol{\mu}_i) \right). \tag{27}$$

Substituting in to (26) and using the facts $\langle (r_i^k - \mu_i^k)(r_i^l - \mu_i^l)\rangle_{Q(R')} = (\Omega_i)_{kl}$ and $\langle (\mathbf{r}_i - \mathbf{r}_j)\Sigma_{ij}^{-1}(\mathbf{r}_i - \mathbf{r}_j)\rangle_{Q(R')} = \langle (\mathbf{r}_i - \boldsymbol{\mu}_i)^T\Sigma_{ij}^{-1}(\mathbf{r}_i - \boldsymbol{\mu}_i)\rangle_{Q(R')} + \langle (\mathbf{r}_j - \boldsymbol{\mu}_i)^T\Sigma_{ij}^{-1}(\mathbf{r}_j - \boldsymbol{\mu}_i)\rangle_{Q(R')}$ we get

$$L_R = -\sum_{ij} \alpha_{ij} \left[\frac{1}{2} \log \frac{1}{|\Omega_i|} + \frac{1}{2}\langle (\mathbf{r}_j - \boldsymbol{\mu})^T\Sigma_{ij}^{-1}(\mathbf{r}_j - \boldsymbol{\mu}_i)\rangle_{Q(R')} + \frac{1}{2}Tr(\Sigma_{ij}^{-1}\Omega_i) \right]. \tag{28}$$

Performing the final expectation gives us

$$L_R = -\sum_{ij} \alpha_{ij} \left[\frac{1}{2} \log \frac{1}{|\Omega_i|} + \frac{1}{2}(\boldsymbol{\mu}_j - \boldsymbol{\mu}_i)^T\Sigma_{ij}^{-1}(\boldsymbol{\mu}_j - \boldsymbol{\mu}_i) + \frac{1}{2}Tr(\Sigma_{ij}^{-1}(\Omega_i + \Omega_j)) \right]. \tag{29}$$

First we differentiate this L_R with respect to each element μ_i^k and equate to zero to get

$$\boldsymbol{\mu}_i = \sum_j \alpha_{ji}(\Sigma_{ji})^{-1}\boldsymbol{\mu}_j + \sum_j \alpha_{ij}(\Sigma_{ij})^{-1}\boldsymbol{\mu}_j. \tag{30}$$

Next we need to optimise the elements of Ω_i. This is less straightforward because these elements occur in determinants and traces, and are subject to a constraint of positive definiteness.

At this stage we will consider the case where both Σ and Ω are diagonal. Then when we take derivatives with respect to $(\Omega_i)_{pp}$ we get

$$(\Omega_i)_{pp} = \frac{1}{\sum_j \alpha_{ij}(\Sigma_{ij})_{pp}^{-1} + \alpha_{ji}(\Sigma_{ji})_{pp}^{-1}}. \tag{31}$$

This gives us the last of the mean field equations, and so we can now perform the optimisation.

A.3 Optimisation of Q(Z)

The terms of the variational log likelihood relevant to this optimisation is given by

$$L_Z = -\Big\langle \log \frac{Q(Z)}{P(Z)} \Big\rangle_{Q(Z)} - \Big\langle \log \frac{Q(X|Z)}{P(X|Z)} \Big\rangle_{Q(Z)Q(X|Z)} - \Big\langle \log \frac{Q(R')}{P(R|Z)} \Big\rangle_{Q(Z)Q(R')}. \tag{32}$$

Using the approximation form $Q(Z) = \sum_{ij} \alpha_{ij}^{z_{ij}}$, substituting in for $Q(X|Z)$, $Q(R)$, $P(Z)$, $P(X|Z)$, $P(R|Z)$ and taking expectations over X and R and Z gives

$$L_Z = -\sum_{ij} \alpha_{ij} \left[\log \frac{\alpha_{ij}}{\gamma_{ij}} + \Psi_{ij} + \Phi_{ij} \right] \tag{33}$$

$$\begin{aligned} \text{where} \quad \Psi_{ij} &= \sum_l m_j^l [\log \sum_k P_{ij}^{kl} \lambda_i^k] \qquad \text{and} \\ \Phi_{ij} &= \frac{1}{2}(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)^T \Sigma_{ij}^{-1} (\boldsymbol{\mu}_i - \boldsymbol{\mu}_j) + \frac{1}{2}\mathrm{Tr}(\Sigma_{ij}^{-1}\Omega_i) + \frac{1}{2}\mathrm{Tr}(\Sigma_{ij}^{-1}\Omega_j). \end{aligned}$$

Differentiating with respect to α_{ij} (with Lagrange multipliers to encode the constraint $\sum_j \alpha_{ij} = 1$) gives the required result $\alpha_{ij} \propto \gamma_{ij} \exp(-\Psi_{ij}) \exp(-\Phi_{ij})$ where the constant of proportionality is given by normalisation.

B The dynamic tree within the PEDT formalism

It is possible to code and run the dynamic tree within the PEDT formalism. We show in this section that the DT is equivalent to the PEDT with a fixed $Q(R')$ up to an additive constant in the variational log likelihood. Suppose we choose a distribution $Q(R')$ to place each node centre in a quadtree like position. Hence the top layer node will be centred on the image, the four nodes in the next layer will be centred in each of the four quadrants of the image, the sixteen nodes of the next layer in the quadrants of those quadrants etc. For non-square images and other node numbers per layer a simple generalisation of this approach can be used distributing nodes evenly across image space. Now we give a variance to each node which will denote the affinities of the dynamic tree (see [29] for an understanding of affinities). This is similar to the method for calculating affinities used in [26]. Then we can set the prior $P^*(Z)$ over connectivity in the dynamic tree to be given by

$$\mathrm{argmin}_{Q(Z)} \int dR' \sum_Z Q(Z)Q(R') \log \frac{Q(Z)Q(R')}{P(Z,R)} \tag{34}$$

where $P(Z)$ and $P(R)$ are the usual PEDT distributions and $Q(Z)$ and $Q(R')$ take the usual form. This gives

$$P^*(Z) = \frac{1}{\Lambda} \exp\left(\int dR' Q(R') \log \frac{P(Z,R)}{Q(R')} \right). \tag{35}$$

where Λ is a normalisation constant. This probability is highest if Z is such that the parents of each node are the closest node from the layer above. Furthermore the distribution $P^*(Z)$ factorises for each node because that is the required form of $Q(Z)$.

The structured variational free energy for the dynamic tree can be used to obtain a lower bound to the probability of the data

$$\log P^*(Y|\Theta) \geq L_{DT} \quad \text{where} \quad L_{DT} = -\sum_{X,Z} Q(Z)Q(X|Z) \log \frac{Q(Z)Q(X|Z)}{P^*(Z)P(X|Z)}. \tag{36}$$

Substituting in for the derived prior (35) gives

$$L_{DT} = -\int dR' \sum_{X,Z} Q(Z)Q(R')Q(X|Z) \log \frac{Q(Z)Q(X|Z)Q(R')}{P(Z)P(R|Z)P(X|Z)} - \log \Lambda. \tag{37}$$

The first term on the RHS is the variational log likelihood for the PEDT. The only annoyance here is the $\log \Lambda$, which is an ignorable constant. Hence we can optimise the variational log likelihood of the dynamic tree by optimising the variational log likelihood of the PEDT, but keeping $Q(R)$ fixed.

References

[1] N. J. Adams. *Dynamic Trees: A Hierarchical Probabilistic Approach to Image Modelling.* PhD thesis, Division of Informatics, University of Edinburgh, 5 Forrest Hill, Edinburgh, EH1 2QL, UK, 2001. Forthcoming.

[2] N. J. Adams, A. J. Storkey, Z. Ghahramani, and C. K. I. Williams. MFDTs: Mean field dynamic trees. Proceedings of the 15th International Conference on Pattern Recognition,

2000.

[3] M. Basseville, A. Benveniste, K. C. Chou, S. A. Golden, R. Nikoukhah, and A. S. Willsky. Modelling and estimation of multiresolution stochastic processes. *IEEE Transactions on Information Theory*, 38:766–784, 1992.

[4] J. Besag. On the statistical analysis of dirty pirtures. *Journal of Royal Statistics, Soc. B*, 48(3):259–302, 1974.

[5] C. A. Bouman and M. Shapiro. A multiscale random field model for Bayesian image segmentation. *IEEE Transactions on Image Processing*, 3(2):162–177, 1994.

[6] E. Charniak. *Statistical Language Learning.* MIT Press, Cambridge, Massachusetts, 1993.

[7] M. Crouse, R. Nowak, and R. Baraniuk. Wavelet-based statistical signal proccessing using hidden Markov models. *IEEE Transactions on Signal Processing*, 46:886–902, 1998.

[8] P. Dayan, G. E. Hinton, R. M. Neal, and R. S. Zemel. The Helmholtz Machine. *Neural Computation*, 7(5):889–904, 1995.

[9] J. S. de Bonet and P. A. Viola. A Non-Parametric Multi-Scale Statistical Model for Natural Images. In M. I. Jordan, M. J. Kearns, and S. A. Solla, editors, *Advances in Neural Information Processing Systems 10*, pages 773–779. MIT Press, Cambridge, MA, 1998.

[10] X. Feng, C. K. I. Williams, and S. N. Felderhof. Combining belief networks and neural networks for scene segmentation. Accepted for publication in IEEE Pattern Analysis and Machine Intelligence, 2001.

[11] D. Geiger and D. Heckerman. Knowledge representation and inference in similarity networks and Bayesian multinets. *Artificial Intelligence*, 82:45–74, 1996.

[12] S. Geman and D. Geman. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Trans. Pattern Analysis and Machine Intellligence*, 6(6):721–741, 1984.

[13] Z. Ghahramani and M. J. Beal. Propagation algorithms for variational bayesian learning. In *Advances in Neural Information Processing Systems 13*. MIT Press, Cambridge MA, 2000.

[14] W. R. Gilks, S. Richardson, and D. J. Spiegelhalter. *Markov Chain Monte Carlo in Practice*. Chapman and Hall, London, 1996.

[15] G. E. Hinton, Z. Ghahramani, and Y. W. Teh. Learning to Parse Images. In S. A. Solla, T. K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 463–469. MIT Press, Cambridge, MA, 2000.

[16] J. Kittler. Statistical pattern recognition in image analysis. In K. V. Mardia, editor, *Statistics and Images 2*, pages 61–75. Carfax Publishing Co., 1994.

[17] S. L. Lauritzen. *Graphical Models*. Oxford University Press, 1996.

[18] H. Lucke. Bayesian Belief Networks as a tool for stochastic parsing. *Speech Communication*, 16:89–118, 1995.

[19] M. R. Luettgen, W. Karl, and A. S. Willsky. Efficient multiscale regularization with applications to the computation of optical flow. *IEEE Trans. Image Processing*, 3:41–64, 1994.

[20] M. R. Luettgen and A. S. Willsky. Likelihood Calculation for a Class of Multiscale Stochastic Models, with Application to Texture Discrimination. *IEEE Trans. Image Processing*, 4(2):194–207, 1995.

[21] A. Montanvert, P. Meer, and A. Rosenfeld. Hierarchical Image Analysis Using Irregular Tesselations. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(4):307–316, 1991.

[22] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[23] P. Pérez, A. Chardin, and J-M. Laferté. Noniterative manipulation of discrete energy-based models for image analysis. *Pattern Recognition*, 33(4):573–586, 2000.

[24] O. Ronen, J. R. Rohlicek, and M. Ostendorf. Parameter Estimation of Dependence Tree Models Using the EM Algorithm. *IEEE Signal Processing Letters*, 2(8):157–159, 1995.

[25] L. K. Saul and M .I. Jordan. Exploiting tractable substructures in intractable networks. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*. MIT Press, Cambridge, MA, 1996.

[26] A. J. Storkey. Dynamic trees: A structured variational method giving efficient propagation rules. In C. Boutilier and M. Goldszmidt, editors, *Uncertainty in Artificial Intelligence*, pages 566–573. Morgan Kauffmann, 2000.

[27] C. von der Malsburg. The correlation theory of brain function. Internal Report 81-2, Max-Planck-Institut für Biophysikalische Chemie, 1981. Reprinted in *Models of Neural Networks*, eds. K. Schulten and H.-J. van Hemmen, 2nd. ed, Springer, 1994.

[28] C. von der Malsburg. Dynamic link architecture. In M. A. Arbib, editor, *Handbook of Brain Theory and Neural Networks*, pages 329–331. MIT Press, 1995.

[29] C. K. I. Williams and N. J. Adams. DTs: Dynamic trees. In M. J. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems 11*. MIT Press, 1999.

[30] C. K. I. Williams and X. Feng. Combining neural networks and belief networks for image segmentation. In T. Constantinides, S-Y. Kung, M. Niranjan, and E. Wilson, editors, *Neural Networks for Signal Processing VIII*. IEEE, 1998.